Claims

1. (Currently Amended) A method of transforming application scenarios written with ontology expressions into software applications, comprising: translating business rules, captured as ontology expressions in a knowledgebase, into an application scenario; transforming the actions in said application scenario into interactions with at least one of: a knowledgebase, one or more presentation components, and underlying services.

2. **(Currently Amended)** The method of claim 1, further comprising at least one of:

- Separation of software in the application business layer, which describes business rules as ontology expressions and service calls, and the application service layer, which describes services;
- Creation and modification of business rules and scenarios that comprise the application business layer at run-time;
- Analysis of successful scenario execution including at least one of: history of successes, history of interpretation failures, learning scenarios that prompt an agent (a user or a program) to re-define the input or to provide more details for better interpretation;

- A queue of scenarios with un-answered questions to resolve unsuccessful interpretations;

-.A system for storing and executing application scenarios, comprising: a knowledgebase containing knowledge facts and business rules captured as ontological expressions, as well as sets of said business rules called application scenarios;

- A Scenario Player that is able to transform actions captured in application scenarios into interactions with at least one of a knowledgebase, presentation components, and underlying services.

3. **(Currently Amended)** The method and system of claim 2, wherein application scenarios are written with the Application Scenario Language, which describes application flow as a set of scenarios that define interactions between system components and agents, where said agents can be users, human beings, or programs.

4. **(Currently Amended)** The system of claim 3, wherein application scenarios consist of scenario actions that can define any of:

(i) interactions between system components and agents

(ii) prompt messages to agents, expected agent responses, if any, and rules for interpretation of agent responses

(iii) invocations of knowledgebase services, like queries, assertions, etc. using service names and optional arguments

(iv) invocations of application services using service and operation names and optional arguments

(v) variables that are replaced with their values at run-time

(vi) conditions based on run-time variable values and knowledgebase queries

(vii) the order of execution of scenarios and scenario acts

(viii) aliases, or multiple ways of expressing the same meaning

(ix) translation policies related to input

(x) possible agent events and event handling rules

(xi) instructions for presentation components

5. **(Currently Amended)** The system of claim 4, further comprising a Presenter component that includes one or more of:

(i) Formatter that prepares data for audio or video presentation or for communication with other programs

(ii) Performer that uses formatted data for actual presentation via voice or screen

(iii) Special engines such as speech, handwriting, or image recognition, which can target specific types of user input.

6. **(Currently Amended)** The method of claim 2, further comprising: awareness of and interaction with knowledgebase and service elements existing on

distributed network systems built with this architecture; at least one method enabling collaboration and sharing of knowledge and service resources over distributed networks.

7. **(Currently Amended)** The method of claim 2, further comprising: translation of existing application scenarios into source code for traditional fixed-form applications with better performance but less flexibility.

8. **(Currently Amended)** The system of claim 4, further comprising a Service Connector component, which includes:

(i) Object Retrieval that is able to find an existing **service o**r load the requested service definition and instantiate the service object at run-time

(ii) Object Registry that associates service objects with service and object names, stores service objects, and makes them reusable

(iii) Method Retrieval that retrieves the proper service method belonging to a selected service object based on the provided method arguments

(iv) Method Performer that performs the requested service operation on the selected service object

9. **(Currently Amended)** The system of claim 4, further comprising an Optimizer component that takes a snapshot of existing rules and scenarios and translates

them into source code in a language such as Java or C#, which can later be compiled into binary code to fix the current application rules into a regular application that lacks flexibility but provides better performance.

10. **(Currently Amended)** The system of claim 4, wherein the Application Scenario Player contains but is not limited to at least one of

(i) Input Type Checker that checks current input and, depending on its type, submits the input to one of several interpreters

(ii) One or more interpreters that translate acts of scenarios or any other input into a direct action by one of the system components, based on scenario and knowledgebase rules

(iii) Queue of Scenarios that stores the current scenario when it cannot be executed at the current time, but needs to be executed later.

(iv) Success Analysis component that can store and retrieve a history of interpretation successes and failures and, in the case of interpretation failure, can invoke one of several learning scenarios that prompt an agent (a user or a program) to re-define the input or to provide more details for a better interpretation.

11. **(Currently Amended)** The system of claim 4, wherein knowledge facts, service, and scenario components are endowed with usage and value properties.

12. **(Currently Amended)** The system of claim 10, wherein a Success Analysis component maintains and consistently refines a list of previously used services with their APIs, keywords, descriptions, and related scenarios in the knowledgebase, and re-evaluates the usage and value properties of the services.

13. **(Currently Amended)** The system of claim 12, wherein a New Agent Request interpreter uses the list of previously used services with their interface definitions, keywords, descriptions, and related scenarios to automatically translate user requests into service interfaces and scenario actions.

14. **(Currently Amended)** The system of claim 13, wherein the New Agent Request interpreter uses a list of previously used services with their interfaces, keywords, descriptions, and related scenarios to offer selected parts of this information to the user for semi-automated translation of new requests into service interfaces and scenario actions.

15. **(Currently Amended)** The system of claim 4, further comprising a Communicator that provides formatted data communications via peer-to-peer distributed networks or any other protocols to provide collaborative access to

knowledge and services existing on distributed network systems built with this architecture

16. **(Currently Amended)** The system of claim 15, wherein the Success Analysis component propagates via the Communicator to distributed network systems information on a new service API or a new knowledge subject after the first success operation that included the service or the knowledge subject and then after each update provided locally by the Success Analysis component.

17. **(New)** The system of claim 8, further comprising a Knowledge Service component, which serves as an adapter to the knowledgebase, adapting the knowledge engine interface to the interface required by the Service Connector.

18. **(New)** The method of claim 3, further comprising: creation and modification knowledge facts, services, and composite service scenarios

19. **(New)** The method of claim 3, further comprising: initial definition of the usage and value properties of newly created facts, services, and scenarios and ability to change these values at run time

20. **(New)** The method of claim 3, further comprising: translation of existing application scenarios into source code for traditional fixed-form applications with better performance but less flexibility.

Signature: Yefim Zhuk / Yefim (Jeff) Zhuk